*Capítulo 2*

# Concepção

A FASE DE CONCEPÇÃO CONSISTE EM UMA ETAPA na qual o analista vai buscar as primeiras informações sobre o sistema a ser desenvolvido. Nessa etapa, assume-se pouco conhecimento do analista sobre o sistema e uma grande interação com o usuário e cliente.

Os artefatos dessa fase são ainda desestruturados, isto é, não são necessariamente completos e organizados. O objetivo é descobrir se vale a pena fazer a análise, mas sem fazer a análise propriamente dita.

Sugere-se que a fase de concepção não dure mais do que umas poucas semanas. Neste período, devem-se levantar todas as informações possíveis com os usuários e clientes que possam vir a ser úteis nas fases posteriores.

Espera-se que, no final dessa fase, o analista tenha uma idéia aproximada do esforço necessário para desenvolver o software.

A fase de concepção corresponde ao primeiro contato do analista com o cliente, no qual o analista vai descobrir o que o cliente quer. Essa fase tem de ser rápida e deve produzir um relatório sucinto do problema que informe, entre outras coisas, se vale a pena continuar trabalhando no problema, se há condições técnicas, financeiras, de cronograma etc., para se levar o projeto ao final. Não se deve investir tempo e dinheiro em um projeto que não se possa concluir.

A maioria dos projetos exige que o analista responda primeiro qual é a visão da empresa para o projeto, ou seja, o que a empresa quer com o projeto,



por que ele está sendo proposto e por que a empresa vai gastar dinheiro com ele. Dentre outras coisas o analista deve responder às seguintes perguntas: o projeto é realizável? A equipe de desenvolvimento tem condições de realizar o projeto? O cliente tem dinheiro para pagar o desenvolvimento? Há tempo disponível?

Essas são as questões que devem ser trabalhadas no primeiro momento. Nessa fase, também surge uma outra questão que muitas vezes os analistas esquecem: comprar ou construir? Muitas vezes, o produto que o cliente quer já está pronto, podendo-se comprar um pacote e adaptá-lo às necessidades da empresa, em vez de construir a partir do zero. Essa questão também deve ser analisada na fase de concepção.

Uma estimativa de custo também é necessária. Afinal, um dos objetivos da fase de concepção é produzir um contrato de desenvolvimento. Nele, deve ser apresentado um cronograma e o custo do projeto. O cliente deve ter uma idéia pelo menos aproximada de quanto ele vai desembolsar até o final do projeto. Vai custar dez mil ou um milhão?

Todas essas questões devem ser respondidas em um tempo relativamente curto. Por isso, sugere-se que a fase de concepção não dure muito. Por que isso? Porque, nessa fase, em geral, o analista e o cliente ainda não têm um contrato fechado, e ela é, do ponto de vista do analista, um investimento no futuro.

As atividades da fase de concepção serão apresentadas neste livro em três partes:

a) Levantamento de requisitos.
b) Organização dos requisitos.
c) Planejamento do desenvolvimento.

A etapa de levantamento de requisitos corresponde a buscar junto ao usuário, seus sistemas e documentos, todas as informações possíveis sobre as funções que o sistema deve executar e as restrições sob as quais o sistema deve operar. O produto dessa fase será o documento de requisitos, primeiro componente do anteprojeto do software.

A etapa de organização dos requisitos serve para estruturar os requisitos para que possam ser abordados nos ciclos de desenvolvimento. Grande parte

dos requisitos funcionais será acomodada em processos de negócio conhecidos como casos de uso. Outros requisitos poderão ser associados a operações simples (como cadastros), outros ainda serão meramente consultas. Os casos de uso, cadastros e consultas serão abordados nos diferentes ciclos, priorizando-se os elementos mais críticos (normalmente casos de uso), e deixando-se para o final os mais elementares (cadastros e consultas).

A etapa de planejamento do desenvolvimento consiste justamente em acomodar os diferentes casos de uso, cadastros e consultas nos ciclos, prever a duração dos ciclos, determinar o tamanho da equipe etc. O produto final será um cronograma financeiro e de atividades para o desenvolvimento do projeto.

## Levantamento de Requisitos

Um sistema a ser analisado é como uma floresta. Para explorar uma floresta desconhecida não é possível em um primeiro momento conhecer cada planta e inseto. Apenas no final da implantação do sistema a equipe poderá dizer que adquiriu conhecimento sobre cada uma das suas diminutas partes. Porém, no primeiro momento do processo de análise, não se pode entrar na floresta para estudar planta por planta, porque assim não se irá adquirir uma visão do todo.

Portanto, a fase de concepção deve fornecer a visão do todo para poder ver o que é mais importante e depois dividir o todo em partes para analisar os detalhes. A organização dos ciclos iterativos corresponde a subdividir a floresta em setores para olhar um setor de cada vez e, assim, poder trabalhar com a complexidade inerente. Um dos objetivos finais da fase de concepção é justamente organizar a divisão do esforço em casos de uso, que serão associados aos diferentes ciclos iterativos.

Na fase de concepção a análise de requisitos é rápida e genérica. Ela é feita em extensão e não em profundidade. O analista deve entender a extensão do que o sistema deve fazer, mas sem detalhar como ele vai fazer. Somente nos ciclos iterativos a análise será aprofundada.

A fase de levantamento de requisitos leva o analista a produzir alguns artefatos (documentos), como os seguintes:





a) A visão geral do sistema, ou sumário executivo, que é um texto corrido, sem necessidade de nenhuma estrutura especial. Ele deve descrever as principais idéias do cliente sobre o sistema.
b) Os requisitos funcionais e não-funcionais, os quais registram todos os tópicos relativos ao que o sistema deve fazer e sob que condições o sistema deve fazer as coisas. Este documento ainda não precisa ser totalmente estruturado, e assume-se que não será completo. Eventuais lacunas desse documento serão preenchidas durante os ciclos iterativos.

Os documentos listados até aqui são fundamentais para a compreensão da fase de concepção do sistema no processo unificado. Existem outros artefatos que também são importantes, mas cujo detalhamento foge ao escopo deste livro, tais quais:

a) O glossário, que é um documento importante porque muitas vezes, quando o analista usa um determinado termo, ele entende uma coisa e o cliente entende outra. Por isso é fundamental que no glossário sejam colocados e definidos todos os termos técnicos do domínio da aplicação. O glossário é, portanto, uma espécie de dicionário da aplicação. O glossário é muitas vezes deixado de lado pelos analistas, mas na maioria das aplicações é importantíssimo definir bem os termos antes de começar a produzir modelos, pois um mesmo nome poderá ter diferentes significados, e nomes diferentes poderão significar a mesma coisa. Por exemplo, em uma videolocadora deve-se dizer que um cliente aluga um filme ou que ele aluga uma fita? Ou esses conceitos são sinônimos? Percebe-se que há uma diferença entre esses dois conceitos quando se conclui que o filme pode estar em DVD ou VHS, e também porque um filme pode ter várias cópias, estando gravado em várias fitas. Então, conceitos que no senso comum são sinônimos na verdade não representam a mesma entidade quando se passa a ter um modelo técnico da informação.
b) Análise de riscos e seu controle. Deve haver um documento, na fase de concepção, em que o analista examina os principais riscos no desenvolvimento, para tentar abordá-los o mais cedo possível nos ciclos iterativos. Assim, os casos de uso com maior risco devem ser resolvidos primeiro.

Isso porque um risco é algo que pode colocar em perigo todo o desenvolvimento, podendo fazê-lo parar. Logo, quanto antes se tratar o risco melhor, visto que paralisar o desenvolvimento de um sistema após um longo período de trabalho é sempre traumático. Muitas vezes não é possível prever se um risco pode ser neutralizado logo no início do desenvolvimento, mas pode-se pelo menos prever a existência de mecanismos de controle.

c) Protótipos e provas. Caso haja necessidade e disponibilidade de tempo, poderão ser desenvolvidos protótipos ou provas do sistema para, juntamente com o cliente, verificar e esclarecer alguns requisitos. Devido à forma como o processo de análise e projeto se desenvolve nas fases seguintes, o protótipo deverá ser do tipo *throw-away*, ou seja, seu código não será usado para desenvolver a aplicação, mas será descartado depois de cumprir sua finalidade, que é a de esclarecer requisitos. Isso é necessário porque a construção da aplicação deverá seguir uma sistemática que vai produzir uma arquitetura bem organizada, tornando inviável tomar um protótipo construído às pressas e sem planejamento como base.

Após analisar essa extensa lista de artefatos, seria natural que surgisse a pergunta: mas isso não é documentação demais? Quando vamos passar realmente ao desenvolvimento do sistema? Na verdade a pergunta tem muita razão de ser, visto que esses artefatos serão sempre opcionais no desenvolvimento de um sistema. A única coisa que é realmente obrigatória, sem a qual não existe sistema, é a produção do código em uma linguagem de programação. O analista deve ser capaz de selecionar os artefatos que serão realmente úteis para o desenvolvimento do seu sistema e ignorar os desnecessários. Afinal, o objetivo não é produzir documentos, mas sistemas implementados. É evidente que alguns documentos ajudam a produzir o sistema com maior eficiência e eficácia, mas deve-se tentar minimizar a produção de papel para evitar a transformação do desenvolvimento em um processo burocrático e ineficiente.

### Visão Geral do Sistema

A visão geral do sistema, ou o sumário executivo, é um documento de texto em formato livre, na qual deve escrever aquilo que conseguiu descobrir de relevante sobre o sistema após as conversas com os clientes e usuários.





Não são propostas regras aqui sobre como deve ser escrito esse documento. Mas sugere-se que não seja longo demais. Uma a duas páginas parecem ser suficientes para descrever de maneira resumida a maioria dos sistemas. Com mais do que isso possivelmente estarão sendo incluídos detalhes não relevantes no resumo e que deveriam ser tratados em outros documentos, como análise de riscos, requisitos ou casos de uso.

Na Figura 2.1 é apresentada a visão geral de um sistema fictício de videolocadora, que será usado como exemplo do processo de desenvolvimento ao longo do livro.

**Sistema de Videolocadora**

*Visão Geral do Sistema*

É proposto o desenvolvimento de um sistema de controle de videolocadora, que vai informatizar as funções de empréstimo, devolução e reserva de fitas. O objetivo do sistema é agilizar o processo de empréstimo e garantir maior segurança, ao mesmo tempo possibilitar um melhor controle das informações por parte da gerência. Deverão ser gerados relatórios de empréstimos por cliente, empréstimos por fita e empréstimos no mês. O sistema deverá calcular automaticamente o valor dos pagamentos a serem efetuados em cada empréstimo, inclusive multas e descontos devidos. A cada devolução de fitas corresponderá um pagamento, não sendo possível trabalhar com sistema de créditos. A impossibilidade de efetuar um pagamento deve deixar o cliente suspenso, ou seja, impossibilitado de tomar emprestadas novas fitas até saldar a dívida.

**Figura 2.1** Sumário executivo do sistema de videolocadora.

Para permitir a apresentação do sistema no curto espaço disponível neste livro, várias simplificações foram consideradas. Por exemplo, apenas fitas VHS são emprestadas. Não há DVDs ou outros tipos de mídia. Além disso, é simplificado o processo de pagamento, que deve ser efetuado na devolução, não sendo possível trabalhar com o sistema de créditos que as videolocadoras usualmente utilizam.

Observe que a visão geral do sistema é apenas uma descrição desestruturada. Existem aqui informações de nível gerencial e de nível operacional. Muitas

vezes, detalhes sobre tecnologias a serem empregados são descritos aqui também (por exemplo: deve-se desenvolver o sistema em Linux).

O analista deve ter em mente que esse documento descreve as principais preocupações do cliente, as quais serão melhor estruturadas nas fases posteriores do processo de análise.

### Requisitos

Existem muitas abordagens sobre análise e engenharia de requisitos. Neste livro procuraremos apresentar apenas um processo prático para obtenção dos requisitos. Sugere-se aos analistas interessados em se aprofundar no assunto que procurem consultar outras obras dentre as muitas disponíveis.

Recomenda-se que os requisitos sejam levantados pela equipe de desenvolvimento em conjunto com os clientes e usuários em uma sessão do tipo *brainstorming*. Esse trabalho certamente deverá ser suportado pela análise de documentos, relatórios e formulários atualmente existentes na empresa.

Os requisitos podem ser classificados em duas grandes categorias:

a) Os requisitos funcionais correspondem à listagem de tudo que o sistema deve fazer.
b) Os requisitos não-funcionais são restrições colocadas sobre como o sistema deve realizar seus requisitos funcionais.

Os requisitos funcionais podem ser ainda classificados em dois grupos:

a) Os requisitos funcionais evidentes, que são efetuados com conhecimento do usuário. Esses requisitos usualmente corresponderão a eventos do sistema e respostas do sistema, ou seja, quaisquer trocas de informação que ocorram pela interface do sistema com o meio exterior.
b) Os requisitos funcionais ocultos, que são efetuados pelo sistema sem o conhecimento explícito do usuário.

Os requisitos não-funcionais podem ser classificados em obrigatórios e desejados, isto é, aqueles que devem ser obtidos de qualquer maneira e aqueles





que podem ser obtidos caso isso não cause maiores transtornos no processo de desenvolvimento.

Além disso, os requisitos não-funcionais podem ser classificados por atributo: se são requisitos de interface, de implementação, de eficiência, de tolerância a falhas etc.

Ainda em relação aos requisitos não-funcionais, existem aqueles diretamente associados a uma função e outros que são gerais para o sistema. Normalmente será útil ter duas tabelas: uma que relacione os requisitos funcionais e suas restrições associadas, e outra que relacione os requisitos não-funcionais gerais, também chamados de requisitos suplementares.

Uma última classificação útil para os requisitos não-funcionais indicará se são permanentes ou transitórios. O requisito permanente nunca mudará com o tempo (por exemplo, a interface feita por meio de janelas), já o requisito transitório poderá sofrer alterações no futuro (por exemplo, a forma de calcular impostos).

Uma possível estrutura para essa tabela de requisitos poderia ter os seguintes campos:

a) Código do requisito funcional (sugere-se "F" seguido de um número. Exemplos: F1, F2, F3, ...).
b) Nome do requisito funcional (especificação curta).
c) Descrição (especificação longa e detalhamento do requisito).
d) Categoria funcional: evidente ou oculto.

Os campos citados correspondem à especificação do requisito funcional. Adicionalmente a cada requisito funcional poderá haver uma lista de requisitos não-funcionais associados, dando origem a uma continuação dessa tabela, que poderia conter os seguintes campos:

a) Código do requisito não-funcional (sugere-se "NF" seguido do número do requisito funcional, seguido de um ponto e o número do requisito não-funcional. Exemplos: NF1.1, NF1.2, ... NF2.1, NF2.2, ...).
b) Nome do requisito não-funcional (especificação curta).
c) Restrição: especificação (longa) do requisito não-funcional.

d) Categoria: tipo de restrição: segurança, performance, compatibilidade etc.
e) Obrigatoriedade: se o requisito é desejável ou obrigatório.
f) Permanência: se o requisito é permanente ou transitório.

Esses mesmos campos poderão ser usados para definir os requisitos suplementares. A única diferença estaria na numeração, já que os requisitos suplementares não estão associados a requisitos funcionais. A numeração poderia ser feita com o prefixo S, seguida de um número de ordem (ex.: S1, S2, ...).

As Figuras 2.2 e 2.3 apresentam uma sugestão para a formatação do documento de requisitos e um exemplo de seu preenchimento.

Deve-se tomar um certo cuidado no enunciado dos requisitos suplementares. Um requisito como "o sistema deve ser fácil de usar" é muito pouco esclarecedor para ser útil. Melhor seria dizer "o sistema terá ajuda on-line em todas as telas e para todas as funções". Isso dá uma idéia mais precisa sobre o que realmente deve ser realizado pelo sistema.

A finalidade principal das classificações de requisitos em tipos e subtipos é a organização. Os requisitos não-funcionais podem ser classificados em várias

| F1 Registrar empréstimos | | **Oculto ( )** | | |
|---|---|---|---|---|
| **Descrição:** O sistema deve registrar empréstimos de fitas, indicando o cliente e as fitas que foram emprestadas, bem como a data do empréstimo e o valor previsto para pagamento na devolução. | | | | |
| ***Requisitos Não-Funcionais*** | | | | |
| **Nome** | **Restrição** | **Categoria** | **Desejável** | **Permanente** |
| NF1.1 Controle de Acesso | A função só pode ser acessada por usuário com perfil de operador ou superior. | Segurança | ( ) | (x) |
| NF1.2 Identificação de Fitas | As fitas devem ser identificadas por um código de barras. | Interface | ( ) | (x) |
| NF1.3 Identificação do Cliente | O cliente deverá ser identificado a partir de seu nome. | Interface | ( ) | ( ) |
| NF1.4 Tempo de Registro | O tempo para registro de cada fita deve ser inferior a um segundo. | Performance | (x) | ( ) |
| NF1.5 Janela Única | Todas as funções relacionadas a empréstimos devem ser efetuadas em uma única janela. | Interface | (x) | (x) |
| ... | ... | ... | ... | ... |

| F2 Calcular descontos | | **Oculto ( x )** | | |
|---|---|---|---|---|
| **Descrição:** O sistema deve calcular descontos nos empréstimos em função da política da empresa. | | | | |
| ***Requisitos Não-Funcionais*** | | | | |
| **Nome** | **Restrição** | **Categoria** | **Desejável** | **Permanente** |
| NF2.1 Desconto de fim de semana | Nos fins de semana, usuários que levam 4 fitas pagam apenas 3. | Especificação | ( ) | ( ) |
| ... | ... | ... | ... | ... |

**Figura 2.2**: Exemplo de tabelas que especificam requisitos funcionais e não-funcionais associados.





| Nome | Restrição | Categoria | Desejável | Permanente |
|---|---|---|---|---|
| S1 Tipo de Interface | As interfaces do sistema devem ser implementadas como formulários acessíveis em um browser HTML. | Interface | ( ) | ( ) |
| S2 Armazenamento de Dados | A camada de persistência deve ser implementada de forma que diferentes tecnologias de bancos de dados possam vir a ser utilizadas no futuro. | Persistência | ( ) | (x) |
| S3 Perfis de Usuário | Os perfis de usuário para acesso ao sistema são:<br>3. Administrador — pode efetuar todas as operações.<br>2. Operador — pode efetuar as operações de empréstimo, devolução, pagamento e cadastramento.<br>1. Convidado — pode efetuar apenas consultas nos próprios dados (cliente). | Segurança | ( ) | ( ) |
| ... | ... | ... | ... | ... |

**Figura 2.3** Exemplo de tabela que especifica requisitos suplementares.

categorias, conforme já visto. Algumas sugestões de possíveis categorias são listadas a seguir:

a) Usabilidade: quais fatores humanos estão envolvidos no sistema? Que tipos de ajuda o sistema vai prover? Quais as formas de documentação ou manuais disponíveis? Como esses manuais vão ser produzidos? Que tipo de informação eles vão conter? Seria interessante definir esses tópicos na fase de concepção, visto que o contrato com o cliente deveria especificar muitas dessas questões.
b) Confiabilidade: que tipo de tratamento de falhas o sistema vai ter? O analista não é obrigado a produzir um sistema totalmente tolerante a falhas, mas deve estabelecer que tipo de falhas o sistema será capaz de gerenciar: falta de energia, falha de comunicação, falha na mídia de gravação etc. Não se deve confundir aqui falha com erro de programação, pois erros de programação são elementos que nenhum software deveria conter. As falhas são situações anormais que podem ocorrer mesmo para um software implementado sem nenhum erro de programação.
c) Performance: que tipo de eficiência e precisão o sistema será capaz de apresentar? Pode-se estabelecer, por exemplo, como requisito de eficiência, que nenhuma consulta à base de dados de clientes vai demorar mais de cinco segundos. Note que na fase de concepção não se define como o sistema fará para cumprir o requisito, apenas se diz que de alguma forma ele terá

de ser cumprido no projeto. Cabe ao projetista e ao programador garantir que o requisito seja satisfeito. Se o analista por algum motivo conclui que o requisito não pode ser cumprido, então o requisito passa a ser um risco do sistema e eventualmente necessitará de um estudo ainda mais aprofundado na fase de concepção, para verificar a possibilidade de sua realização.

d) Configurabilidade: o que pode ser configurado no sistema? Devem-se definir os elementos que poderão ser configurados pelo usuário sem a necessidade de recompilar o sistema. Exemplos de itens configuráveis são: o tipo de modem, impressoras, a moeda do país, políticas da empresa etc.
e) Segurança: quais são os tipos de usuários e que funções cada um pode executar? Sugere-se definir os perfis de usuários como um requisito suplementar e adicionar um requisito não-funcional de "controle de acesso" a todos os requisitos funcionais evidentes, para indicar como o acesso às funções do sistema é controlado.
f) Implementação: qual linguagem deve ser usada? Por que motivo? Que bibliotecas estarão disponíveis? Quais bancos de dados serão acessíveis?
g) Interface: como deve ser a interface? Vai ser seguida alguma norma ergonômica?
h) Empacotamento: de que forma o software deve ser entregue ao usuário final?
i) Legais: muitas vezes uma equipe de desenvolvimento deve contar com uma assessoria jurídica para saber se está infringindo direitos autorais ou normas específicas da área para a qual o software está sendo desenvolvido.

Embora essa lista seja extensa, o analista deve ter em mente que se trata apenas de um modo de classificar quais são os requisitos relevantes em quais dos tipos listados. Não há necessidade de procurar requisitos que não existem, por exemplo, estabelecendo complicados requisitos de empacotamento para um cliente para o qual não faz a menor diferença a forma como o software será entregue.

A fase de levantamento de requisitos deve ser uma fase de descoberta e não de invenção, na qual a equipe de analistas, junto com o cliente e os usuários, vai procurar listar o maior número possível de capacidades e restrições, mas sem se preocupar também demasiadamente em ter uma lista completa nesta fase, o que em geral é impossível. Os requisitos não descobertos nessa fase





deverão ser convenientemente acomodados ao longo do restante do processo de desenvolvimento.

O documento ou a tabela de requisitos deve registrar as capacidades do sistema e as condições às quais ele deve se conformar. Os desafios ligados a essas informações são os seguintes:

a) Como descobrir os requisitos.
b) Como comunicar os requisitos para as outras fases ou equipes do projeto.
c) Como lembrar os requisitos durante o desenvolvimento e verificar se foram todos atendidos. Não adiantaria escrever um belo documento e depois não ter como saber se os requisitos foram ou não atendidos pelo projeto. É importante dispor de mecanismos para fazer sistematicamente essa verificação.
d) Como gerenciar a mudança dos requisitos.

Posteriormente, quando os casos de uso forem escritos, os requisitos atendidos em cada um deles deverão ser indicados com clareza. Assim, será possível verificar se algum requisito deixou de ser tratado. Dessa forma, a listagem dos casos de uso e o levantamento dos requisitos se complementam: os requisitos são o resultado desestruturado de uma pesquisa, e os casos de uso, de certa maneira, organizam essa informação.

É desejável ter em mente também que os requisitos inevitavelmente mudam durante o desenvolvimento do projeto. Deve-se esperar então poder gerenciar a mudança dos requisitos nas demais fases de desenvolvimento.

Outras vezes, os requisitos mudam depois do desenvolvimento. Podem mudar as condições de contexto, ou a forma de trabalho da empresa, ou as políticas da empresa. Embora essas mudanças não possam ser previstas pelo analista, podem ser criados mecanismos que as acomodem ao sistema quando e se surgirem. Existem padrões de projeto específicos para tratar essas instabilidades do sistema.

De volta ao exemplo da videolocadora, pode-se verificar que as políticas de desconto da empresa são requisitos transitórios: hoje se pratica uma política de locar quatro fitas pelo preço de três. Pode acontecer de, depois de o sistema estar implementado, a empresa decidir eliminar essa política de desconto e

passar a dar desconto de 20% na locação somente para clientes com mais de 60 anos de idade, independente da quantidade de fitas. Algum tempo depois, além da política de descontos para idosos, a empresa pode resolver dar descontos também para clientes que devolvam as fitas antes das 19 horas.

Essas mudanças são totalmente imprevisíveis. Se o sistema não estiver estruturado para acomodar mudanças nos requisitos, haverá excesso de trabalho para implementá-las.

Esse tipo de situação faz com que os processos de análise e projeto dirigidos por requisitos (Alford, 1991) sejam inadequados para muitos sistemas. Fundamentar a estrutura de um sistema em seus requisitos é como construir um prédio sobre areia movediça. Quando os requisitos mudam, a estrutura do sistema muda. A análise e o projeto orientados a objetos, porém, propõem que a estrutura do sistema seja fundamentada em elementos muito mais estáveis: as classes de objetos, que representam a informação gerenciada pelo sistema.

Observe que, em contraponto aos requisitos transitórios anteriormente descritos, existem requisitos permanentes. Assim, dificilmente um sistema de videolocadora deixará de gerenciar informações sobre clientes, empréstimos, pagamentos, multas etc. Outros elementos poderão ser acrescentados sem provocar alteração nos elementos já existentes. Esses elementos, que constituem a informação a ser gerenciada, serão os constituintes básicos da arquitetura do sistema (classes). Os requisitos serão acomodados nesses elementos seguindo o princípio aberto-fechado (Meyer, 1988), segundo o qual uma classe está sempre pronta para uso (fechada em sua funcionalidade), mas aberta para extensões, ou seja, para eventualmente acomodar novos requisitos.

## Organização dos Requisitos

Uma vez que os requisitos tenham sido levantados, cabe agora organizá-los em grupos correlacionados, de forma a abordá-los nos ciclos iterativos. Os requisitos podem ser agrupados então do seguinte modo:

a) Casos de uso. Os principais processos de negócio da empresa (casos de uso) possivelmente abarcarão um número considerável de requisitos funcionais.





b) Conceitos. Os conceitos envolvidos com o sistema possivelmente sofrerão operações de manutenção ou cadastro. Em geral, essas operações são: inserir, alterar, remover e consultar. Nem sempre essas operações de manutenção de informações vão aparecer nos casos de uso, e nem é necessário que isso aconteça. Basta definir quais são os conceitos (informações) que devem sofrer manutenção e indicar a implementação das operações de cadastro.
c) Consultas. Neste livro o termo "consulta" refere-se tanto a consultas exibidas em tela quanto a relatórios impressos. Consulta é qualquer acesso à informação do sistema que não altera a informação em si. O tipo de informação acessada ou a maneira como ela é exibida não interfere na definição.

Alguns autores sugerem que todas as interações possíveis com o sistema devem ser representadas como casos de uso, inclusive as consultas. Todavia, observa-se que a estrutura de interação de consultas, inserções, exclusões e alteração de dados simples usualmente segue uma mesma forma geral e consiste em um único acesso ao sistema. Assim, não existe muita praticidade em considerar tais operações como casos de uso. O caso de uso vai ser expandido na fase de análise e vai também gerar um diagrama de seqüência para que o analista possa especificar as várias operações executadas dentro do processo. Quando se trata de uma operação simples como "alterar o endereço de um cliente", não há necessidade de produzir tais artefatos.

Sugere-se, para efeito de análise, considerar como caso de uso apenas os processos de negócio que envolvam transações com mais de um passo. Operações típicas de manutenção de cadastro (inserir, alterar, excluir e consultar) ficariam associadas aos conceitos relacionados e seriam especificadas de um jeito mais simples na fase de análise.

### Organização dos Requisitos em Casos de Uso

Na fase de concepção é necessário identificar os grandes processos da empresa. As operações elementares (consultas e alterações de dados) possivelmente estarão inseridas dentro desses processos. Por exemplo, no caso do sistema de videolocadora, os grandes processos são: "fazer empréstimo de fitas",

"fazer devolução de fitas", "fazer reservas" etc. Funções mais simples como "cadastrar cliente" e "pagar despesas" poderão ocorrer dentro dos processos maiores com mais freqüência do que como processos independentes. No caso da locadora, o cadastramento do cliente acontecerá durante o processo de locação de fitas, caso o cliente não possua ainda um cadastro. Já o pagamento ocorrerá nessa locadora apenas como parte do processo de devolução de fitas, já que não foi previsto sistema de crédito para a locadora.

Os grandes processos de negócio da empresa são também chamados de casos de uso. Eles devem cobrir as principais atividades da empresas ligadas ao sistema que será implementado. O objetivo de listar os casos de uso é levantar informações sobre como o sistema interage com possíveis usuários e quais consultas e transformações da informação são necessárias além daquelas já identificadas na fase de levantamento de requisitos.

Cada caso de uso será associado a um conjunto de requisitos funcionais do sistema. Na fase de concepção, os casos de uso só precisam ser listados e eventualmente descritos em alto nível. É ainda interessante indicar os atores envolvidos e os requisitos correlacionados. A Figura 2.4 apresenta uma possível estrutura para a apresentação dos casos de uso na fase de concepção.

| Nome | Atores | Descrição | Referências Cruzadas |
|---|---|---|---|
| Emprestar Fitas | Cliente, Funcionário | O cliente se identifica e identifica as fitas que deseja levar. O funcionário faz o registro e libera as fitas para empréstimo. | F1, F3, F5, F9, F10 |
| Devolver Fitas | Cliente, Funcionário | O cliente entrega ao funcionário as fitas. O funcionário faz o registro da devolução e o cliente efetua o pagamento devido. | F2, F4, F6, F7, F8 |
| Reservar Fitas | Cliente, Funcionário | O cliente solicita a reserva de um ou mais filmes. O funcionário registra a reserva. | F11, F12 |

**Figura 2.4** Listagem de casos de uso da fase de concepção.

Para descobrir os casos de uso, devem-se identificar os atores envolvidos com o sistema (funcionários, gerentes, clientes etc.). Para tanto, o analista deve responder a algumas perguntas, como as seguintes:

a) Quem usa o sistema?
b) Quem mantém ou configura o sistema?





c) Quais outros sistemas de informação utilizam ou são utilizados pelo sistema?
d) Quem busca informações no sistema?
e) Quem provê informações para o sistema?

Deve-se então identificar os principais processos de negócio iniciados ou com participação desses atores. A cada grande processo possivelmente corresponderá um caso de uso.

Existe um diagrama na UML para representar casos de uso e seus atores. A principal utilidade desse diagrama está no fato de se poder associar a ele, caso se utilize uma ferramenta CASE, um conjunto de outros artefatos que representam a interação entre os atores e o sistema. A Figura 2.5 apresenta um diagrama de casos de uso para o sistema de videolocadora. As elipses representam casos de uso, os bonecos representam atores (usuários) e o retângulo representa a fronteira do sistema.

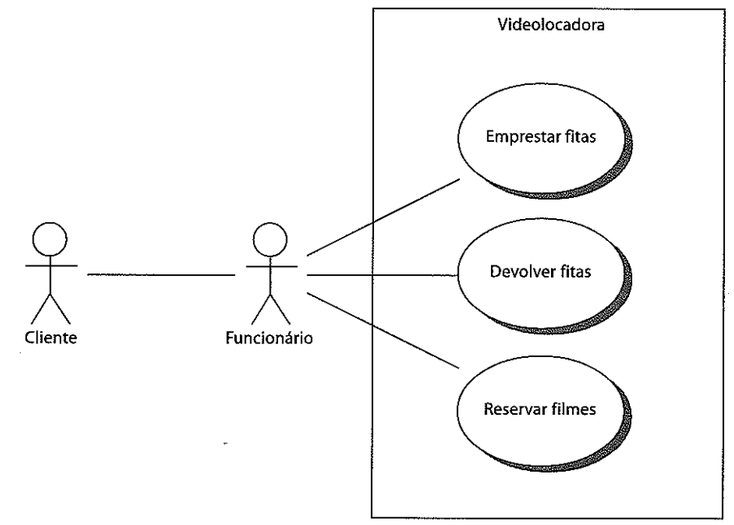


**Figura 2.5** Diagrama de casos de uso da UML.

Para se ter uma noção de granularidade em relação a casos de uso (para não definir processos muito longos ou muito triviais como casos de uso), pode-se levar em conta os seguintes limites:

a) Um caso de uso deve ser monossessão, isto é, executado em uma única interação e não se estendendo ao longo de vários dias.
b) Um caso de uso deve ser interativo, com informações fluindo para dentro e para fora do sistema. Operações elementares como cadastrar um cliente ou alterar informações da fita dificilmente serão casos de uso, pois são efetuadas em um único passo.
c) Um caso de uso deve produzir uma alteração consistente na informação armazenada. Isso exclui a possibilidade de consultas serem casos de uso, pois não alteram a informação, apenas exibem os dados já existentes com alguma formatação específica. Além disso, um fragmento de um caso de uso que não produza uma informação consistente não poderá ser considerado um caso de uso. Deve-se considerar que o caso de uso é uma transação com início e fim bem delimitados. Por "bem delimitado" entenda-se que o usuário poderia ativar o sistema, executar o caso de uso e desligar o sistema em seguida, e a operação estaria completa e consistente. Excluem-se desse processo procedimentos de *login* e *logoff*, que, como será visto, são tratados à parte dos processos de negócio.

Consultas, cadastros e quaisquer outras operações que não estão sendo consideradas como casos de uso, para efeito de estudo, serão agrupados nas duas etapas seguintes da organização dos requisitos.

### Organização dos Requisitos em Função de Conceitos

Algumas operações relativamente simples e elementares (de um único passo), como o registro de uma fita ou de um pagamento, não devem ser consideradas casos de uso por si só, pois não há necessidade de estudar seu processo interativo, que é de um único passo.

Além disso, essas operações elementares possivelmente poderão fazer parte de algum caso de uso propriamente dito. Por exemplo, o cadastro de uma fita fará parte do caso de uso de compra de fitas, e o registro do pagamento de uma locação fará parte do caso de uso de devolução de fitas.





Para identificar corretamente essas operações sem o ônus de incluí-las em um caso de uso arbitrário, sugere-se que o analista procure elaborar um modelo conceitual preliminar na fase de concepção. Haverá um capítulo posterior destinado ao estudo do modelo conceitual. Assim, aqui apenas será indicado que esse modelo ou diagrama deve conter os conceitos e associações que constituem a informação a ser armazenada pelo sistema. A Figura 2.6 apresenta um modelo conceitual simplificado para o problema da videolocadora.

Nessa fase, basta ao analista saber que os conceitos identificados são informações complexas (as quais não correspondem a simples tipos primitivos ou alfanuméricos), localizadas nos textos do documento de requisitos.

Se desejar, o analista pode incluir atributos nos conceitos, indicando informações alfanuméricas associadas, como nome, endereço e telefone (ver classe Cliente na Figura 2.6).

As associações, por sua vez, são ligações estruturais entre conceitos, que complementam a informação registrada por eles.

Não se devem incluir operações (métodos) nesse diagrama, nem direção de navegação nas associações. Haverá um momento próprio na fase de projeto para se trabalhar com esses elementos.

Uma vez identificados os conceitos principais, pode-se definir uma tabela que indicará se eles devem sofrer inserção (I), alteração (A), exclusão (E) ou

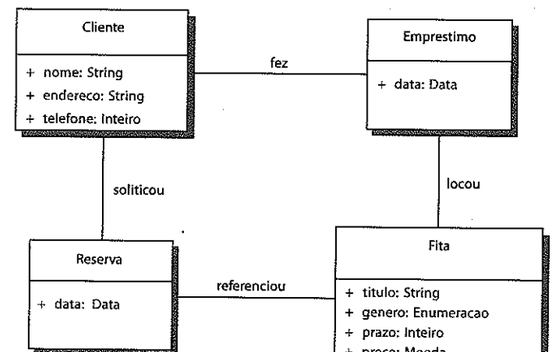


**Figura 2.6** Um modelo conceitual preliminar para a videolocadora.

consulta (C). Pode-se construir uma tabela para indicar os conceitos e quais operações serão possíveis sobre eles (Figura 2.7). Pode-se, nessa tabela, também, identificar restrições ou regras de validação para a execução das operações. Por exemplo, "um cliente só pode ser excluído se não houver nenhum empréstimo alocado a ele."

| Conceito | I | A | E | C | Observação | Ref. Cruzadas |
|---|---|---|---|---|---|---|
| Cliente | x | x | x | x | Só é possível excluir se não houver empréstimos associados | F13 |
| Reserva | x | x | x | x | | F15, F16 |
| Fita | x | x | x | x | Só é possível excluir se não houver empréstimos associados | F18 |
| Empréstimo | | | x | x | A inclusão de empréstimo só pode acontecer através do caso de uso "emprestar fitas".<br>Não é possível alterar um empréstimo, apenas excluir. | F17, F19 |

**Figura 2.7** Listagem de conceitos e operações de manutenção.

Eventualmente, haverá redundância entre algumas operações de manutenção e operações a serem descobertas nos casos de uso. Para evitar que isso se torne um problema, uma sistemática de nomenclatura de operações bastante rígida deve ser usada para evitar a dupla definição de uma mesma operação.

### Organização dos Requisitos em Consultas

A geração de relatórios e consultas é um dos processos mais fundamentais dentro de sistemas de informação. A identificação correta dos relatórios desejados na fase de concepção pode ajudar muito no levantamento dos requisitos e na elaboração do sistema. Contudo, para o usuário, o processo de gerar um relatório usualmente consiste em selecionar alguns parâmetros e enviar o resultado para um dispositivo como uma impressora. O processo basicamente é sempre o mesmo. Então, a geração de relatórios não precisa ser considerada caso de uso do sistema. Basta definir o formato em que a informação deverá ser exibida e definir um modo padrão para o usuário gerar os relatórios. Não será necessário tratá-los como casos de uso porque o resultado será o mesmo todas as vezes, variando apenas os parâmetros e resultados exibidos.

Além disso, um dos princípios de casos de uso é que eles produzem no sistema uma mudança de estado, ou seja, uma modificação na informação armazenada. Assim, quaisquer registros como registro de empréstimo, devolução e reserva poderão ser efetivamente casos de uso, mas consultas e relatórios por definição não geram nova informação no sistema e, portanto, não são considerados casos de uso.





Desse modo os relatórios e consultas só serão listados na fase de concepção, visando a uma construção em etapa posterior. Não é necessário listar aqui as consultas que simplesmente permitem visualizar um conceito armazenado (por exemplo, consultar as informações de um cliente), porque esse tipo de consulta já foi considerado nas operações de manutenção de informações de conceitos na seção anterior. As consultas a serem listadas aqui são do tipo que combina informações de diferentes conceitos, como "relatório mensal de vendas", "total de empréstimos por cliente", "número total de empréstimos de uma fita" etc.

As consultas listadas aqui possivelmente já foram descritas e detalhadas nos requisitos funcionais. Logo, será suficiente, na maioria das vezes, apenas listar as consultas e suas referências cruzadas, como na Figura 2.8.

| Nome | Referências Cruzadas |
|---|---|
| Vendas Mensais | F20, F21, F22 |
| Clientes Suspensos | F13, F23, F1 |
| ... | ... |

**Figura 2.8** Listagem de consultas.

Adicionalmente, o analista poderá anexar desenhos dos relatórios ou consultas indicando como o cliente gostaria de visualizar as informações, se necessário.

### Planejamento dos Ciclos Iterativos

Os ciclos iterativos são miniprojetos que se baseiam em casos de uso, conceitos e consultas. Esses miniprojetos devem ser curtos e com tempo predeterminado.

Se for definido um período de, por exemplo, quatro semanas para um ciclo, isso significa que, nesse tempo, deverá ter sido executada uma análise completa dos casos de uso, conceitos e/ou consultas associados ao ciclo. Essa análise envolve uma série de atividades que serão descritas no próximo capítulo.

Além disso, dentro deste mesmo período deverá ser feito o projeto, a implementação e os testes das partes do sistema referentes aos casos de uso, conceitos e/ou consultas associados ao ciclo.

Recomenda-se que o cronograma tenha uma certa folga, porque vários requisitos novos serão fatalmente descobertos ao longo dos ciclos iterativos e deverão ser acomodados à medida que o desenvolvimento se mantém. Essa folga deve ser determinada com realismo a partir da medida de desempenho que a própria equipe tem em relação à sua capacidade de identificar requisitos corretamente.

Conforme se avança de um ciclo iterativo para o seguinte, espera-se uma menor complexidade de trabalho, porque justamente os casos de uso mais críticos já foram tratados, e eram eles que apresentavam maior risco. Na medida em que se vai avançando de um para outro ciclo, os casos de uso vão se tornando mais simples, e a maior parte do código também já terá sido implementada.

Portanto, ao longo dos ciclos iterativos, o conhecimento sobre o sistema vai aumentando, e grande parte da arquitetura já vai sendo definida e implementada. Dessa forma, os processos de trabalho vão ficando cada vez mais sistemáticos. A sugestão de deixar a geração de relatórios e as consultas do sistema para os últimos ciclos deve-se ao fato de que nessa hora a estrutura geral de informações do sistema já estará pronta, não havendo muito mais a fazer além de formatar as informações para exibir ou imprimir.

### Estimação de Esforço

Para estimar o esforço total de desenvolvimento é possível aplicar medidas de pontos de função (Garmus & Herron, 2000) ou de pontos de caso de uso (Karner, 1993). Este livro não vai se aprofundar no tema, visto que existe muita literatura especializada na área. Apenas é indicado que, se a opção for por pontos de função, deverá ser feita uma associação de esforço a cada um dos requisitos funcionais definidos. Assim, o esforço total de um ciclo será possivelmente o somatório dos pontos de função das referências cruzadas dos casos de uso, conceitos e consultas associados ao ciclo.

A partir do cálculo dos pontos de função será possível saber quanto tempo (em média) cada elemento do sistema levará para ser desenvolvido. Dessa maneira, pode-se alocar um número de desenvolvedores adequado a cada uma das fases.





### Prioridades dos Elementos a Abordar

A fim de tratar os maiores riscos do sistema nas fases iniciais, sugere-se que sejam abordados primeiramente os casos de uso, depois as operações de manutenção de conceitos e, por último, as consultas. Os riscos maiores estarão fatalmente nos processos de negócio; por isso inicia-se com os casos de uso. As operações de cadastro seguem sempre um certo padrão e não apresentam riscos. Por fim, as consultas não alteram dados no sistema, mas apenas apresentam a informação já armazenada em um formato ou tabulação específico e apresentam pouquíssimo risco.

Quando a quantidade de casos de uso identificados é relativamente grande, deve-se proceder ainda a uma priorização do conjunto de casos de uso, atribuindo-se sempre os casos de uso mais importantes aos primeiros ciclos iterativos.

A importância de um caso de uso refere-se a quanto o analista pode aprender com ele. Com certeza, casos de uso como "emprestar fita" e "devolver fita" na videolocadora são muito importantes, porque todo o funcionamento da videolocadora se baseará nesses processos. Já o caso de uso "reservar fita" pode ser visto como de segunda importância, visto que ele não trata da parte mais vital do processo para a empresa.

A identificação dos casos de uso mais importantes permite que eles sejam selecionados para serem estudados nos primeiros ciclos iterativos, deixando para depois os casos de uso mais triviais.

Quando se ordenam os ciclos iterativos pela importância dos casos de uso, os benefícios de trabalhar dessa forma são muitos. Os grandes riscos do desenvolvimento do software são tratados cedo, porque se começa analisando os processos mais importantes. Os processos de negócio da empresa devem ser muito bem compreendidos e modelados; os demais são acessórios.

Em cada ciclo iterativo deve-se trabalhar um subconjunto dos casos de uso, operações de manutenção de informações e/ou consultas. Para o exemplo da videolocadora, o processo de desenvolvimento poderia ser planejado de acordo com a Figura 2.9. O número entre parênteses após cada atividade indica o esforço em horas estimado para realizá-la.

| Ciclo | Casos de Uso | Manutenção de Informações | Consultas | Observações | Esforço estimado |
|---|---|---|---|---|---|
| 1 | Emprestar Fita (550) | - | - | Neste ciclo ainda não será implantado o mecanismo de persistência | 550 horas |
| 2 | Devolver Fita (300) | - | - | Implementar mecanismo de persistência (300 horas) | 600 horas |
| 3 | Reservar Filme (270) | Fita (100), Cliente (100) e Reserva (100) | - | - | 570 horas |
| 4 | - | Emprestimo (100) | todas (400) | - | 500 horas |

**Figura 2.9** Exemplo de planejamento em ciclos iterativos.

A tabela indica, para cada ciclo, quais elementos serão abordados. Pode-se usar o campo de observações para indicar se, por exemplo, alguns aspectos do sistema serão postergados para outro ciclo.

O progresso no desenvolvimento com ciclos iterativos é logo visível, porque em pouco tempo existirá código implementado para que o usuário possa verificar se os seus desejos são atendidos. O usuário se engaja de imediato e participa ativamente ao longo do processo.

A aprendizagem que o analista ou projetista obtém em cada ciclo pode ser metodicamente usada para melhorar o ciclo seguinte. Inclusive se for o caso de procurar obter uma certificação de qualidade de processo de desenvolvimento, como o ISO9000-3 (Kehoe et al., 1996) ou o CMM/Spice (Emam et al., 1998), a possibilidade de usar informações de um ciclo para melhorar o processo nos demais ciclos é um fator muito relevante.

### Cronograma de Execução e Custos

A construção do cronograma de execução dependerá dos seguintes fatores:

a) Tempo total estimado para o projeto (em hora/pessoa).
b) Tempo disponível (em semanas ou meses).
c) Tamanho da equipe.
d) Estruturação da equipe.

Dados esses elementos, o cronograma deve seguir um padrão cíclico. No caso da videolocadora, foram propostos quatro ciclos. Observa-se na Figura 2.9 que todos os ciclos têm um esforço estimado aproximadamente igual e que o ciclo com maior esforço exige cerca de 600 horas de trabalho. A partir da suposição de que a empresa tem à disposição tantos analistas, projetistas e





programadores quantos sejam necessários e se deseja desenvolver esse projeto em cerca de 90 dias úteis, então um possível cronograma poderia ser feito conforme a Figura 2.10. Nesse cronograma supõe-se que a fase de concepção já foi concluída e aloca-se tempo extra para a fase de implantação, a qual ocorrerá no final do projeto.

| Dias: | 1-10 | 11-20 | 21-30 | 31-40 | 41-50 | 51-60 | 61-70 | 71-90 |
|---|---|---|---|---|---|---|---|---|
| Ciclo 1 | análise | projeto | implementação | testes | | | | |
| Ciclo 2 | | análise | projeto | implementação | testes | | | |
| Ciclo 3 | | | análise | projeto | implementação | testes | | |
| Ciclo 4 | | | | análise | projeto | implementação | testes | |
| Implantação | | | | | | | | implantação |

**Figura 2.10** Exemplo de cronograma para o desenvolvimento do sistema de videolocadora.

Como cada dia de trabalho tem cerca de 8 horas, para se chegar às 600 horas necessárias para realizar um ciclo em 40 dias de trabalho (320 horas) será necessário alocar oito profissionais por ciclo (por exemplo, três analistas, dois projetistas, dois programadores e um testador). O número de profissionais alocados por tipo de tarefa dependerá de uma estimativa de complexidade de cada tarefa, bem como de uma estimativa de performance da própria equipe. Em situações reais será verificado que a competência dos profissionais e as características intrínsecas do sistema poderão exigir modificações no planejamento (criando-se, por exemplo, ciclos com mais projetistas do que analistas etc.). Além disso, a relação entre o número de trabalhadores e o tempo para concluir o projeto não é linear. Se um analista leva dez dias para cumprir uma tarefa, isso não quer dizer que dez analistas levarão um dia.

Equipes pequenas terão de adaptar o formato do cronograma à sua capacidade. Por exemplo, se o projeto vai ser desenvolvido por uma equipe de apenas duas pessoas, sendo um analista-projetista e um programador-testador, então o cronograma poderia ser feito como na Figura 2.11.

| Dias: | 1-20 | 21-40 | 41-60 | 61-80 | 81-100 | 101-120 | 121-140 | 141-160 | 161-180 | 181-200 | 201-220 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ciclo 1 | análise | projeto | impl. | testes | | | | | | | |
| Ciclo 2 | | | análise | projeto | impl. | testes | | | | | |
| Ciclo 3 | | | | | análise | projeto | impl. | testes | | | |
| Ciclo 4 | | | | | | | análise | projeto | impl. | testes | |
| Implantação | | | | | | | | | | | implant. |

**Figura 2.11** Um cronograma alternativo para uma equipe de apenas duas pessoas.

Se for considerado que cada ciclo tem 600 horas e apenas duas pessoas vão trabalhar, então serão necessários cerca de 75 dias para concluir um ciclo, isto é, cerca de 20 dias em cada atividade. Assim, essa equipe levará mais do que o dobro do tempo para concluir o trabalho em relação à equipe de 8 pessoas considerada na Figura 2.10.

O custo de um projeto para o cliente é calculado com base na mão-de-obra necessária, no tempo despendido e na infra-estrutura. É preciso alguma experiência para poder dizer quanto tempo vai durar um projeto. Mas o recomendável é que cada ciclo iterativo dure entre duas e oito semanas.

Equipes de desenvolvimento numerosas demandam ciclos mais longos, pois é necessário despender mais tempo coordenando os trabalhos das diversas subequipes. Porém, equipes grandes podem trabalhar vários casos de usos simultaneamente.

### Planejamento Adaptativo e Registro de Pendências

Ao final de cada ciclo iterativo deve haver o replanejamento dos ciclos posteriores, porque a metodologia aqui apresentada não é preditiva, mas adaptativa.

A metodologia seria preditiva se na fase de concepção fosse feito um cronograma detalhado de todo o desenvolvimento. Mas ela é adaptativa no sentido de que o cronograma do desenvolvimento é genérico e, ao final de cada ciclo, o próximo ciclo é replanejado para acomodar os imprevistos. A duração dos ciclos é fixa, mas a atividade que é feita dentro de cada um pode ser adaptada.

É importante que cada equipe registre ao final de cada ciclo iterativo se algum aspecto previsto para o ciclo não foi tratado. Espera-se que alguns ciclos sejam mais complexos do que outros e pendências surjam. A única forma de se conviver com isso é regularizando o registro das pendências para que a equipe retome o trabalho naquele aspecto em um ciclo subseqüente no qual exista mais folga. Não se pode, porém, atrasar a entrega dos artefatos previstos no cronograma, visto que outras equipes estarão aguardando por eles para iniciar seu trabalho.

### Ciclos Iterativos, Subsistemas e Refinamentos Sucessivos

A divisão em ciclos iterativos não é necessariamente uma divisão em subsistemas. Na verdade, se um sistema tem subsistemas, um mesmo ciclo





iterativo pode elaborar casos de uso de diferentes subsistemas, e um subsistema completo poderá ser tratado em diferentes ciclos iterativos.

Assim, os ciclos iterativos não geram necessariamente o código de um subsistema. Na verdade, o sistema como um todo vai crescendo, a cada ciclo iterativo, como uma bolha, que aumenta a cada iteração. Inicialmente, são desenvolvidas as estruturas mais críticas ligadas aos casos de uso considerados mais importantes. A partir daí acrescentam-se as estruturas complementares e acessórias.

A técnica de ciclos iterativos também não consiste em realizar refinamentos sucessivos. Entende-se que refinamentos sucessivos são uma maneira cíclica de trabalhar sobre um sistema no qual se vai compreendendo, projetando e implementando um sistema, primeiro de uma forma mais genérica, e depois detalhando mais a cada passo. A técnica de refinamentos sucessivos tem mais relação com o desenvolvimento funcional *top-down* do que com ciclos iterativos. Em relação a um ciclo iterativo, ele deve tratar todas as questões possíveis sobre o caso de uso, conceito ou consulta, ou seja, vai-se até os últimos detalhes da análise, projeto e programação, sem deixar nada para ser refinado depois, até porque os outros ciclos iterativos vão tratar outras partes do sistema, não retornando a esse caso de uso, exceto quando se identifica que algo na análise ou projeto precisa ser revisado ou complementado (como no primeiro ciclo da Figura 2.9). No entanto, esse retorno não é a regra, como nos refinamentos sucessivos, mas a exceção.

### Casos de Uso e Ciclos Iterativos

A recomendação de abordar inicialmente os casos de uso mais complexos e só depois os cadastros mais simples coloca dúvidas em relação à possibilidade de implementar e testar o código dos casos de uso sem ter ainda a possibilidade de cadastrar as entidades que participam do caso de uso. Por exemplo, como testar o empréstimo de fitas se ainda não foi implementada a operação de cadastramento de fitas? A resposta é simples: para testar o processo de empréstimo basta usar uma bateria de dados sobre fitas inserida manualmente no banco de dados ou, ainda, usar dados gerados a partir de um caso de testes, isto é, um programa que produz os dados a ser usados nos testes durante sua execução, sem a necessidade de o testador entrar com esses dados pela interface do sistema.

Não é necessário, portanto, que as interfaces de cadastramento de fita estejam prontas para ser possível testar as interfaces de empréstimo. O prioritário deveria ser implementar e testar a parte do sistema que faz empréstimos em relação à parte do sistema que faz cadastro de fitas porque o processo de empréstimo, como já visto, apresenta mais riscos em relação ao sistema, e esses riscos devem ser eliminados o quanto antes. Dificilmente, o processo de cadastro de fitas vai produzir alterações significativas no processo de empréstimo, mas o inverso pode ser verdade. Assim, para evitar o retrabalho deve dar prioridade aos casos de uso mais críticos no cronograma de desenvolvimento. A tendência é que as coisas funcionem melhor desse modo, pois se o analista começasse analisando o cadastramento de fitas, sem ter estudado ainda o empréstimo e a devolução de fitas, ele não teria elementos para saber exatamente quais informações a empresa necessita guardar sobre as fitas.

Porém, tudo isso é mais relevante quando se está efetivamente analisando um sistema para o qual não existe uma especificação bem compreendida. Quando a análise envolve situações como reengenharia de sistemas ou engenharia reversa, assume-se que os processos de negócio e suas regras já estão convenientemente representados no sistema antigo. Nessas ocasiões, o uso dos casos de uso como ferramenta de estudo não é tão crítico. E a ordem de abordagem dos casos de uso pode ser substancialmente alterada.

## Exercícios

**1. O que se pode afirmar sobre a fase de concepção?**

a) Tem como objetivo gerar um planejamento detalhado e sistemático para todas as etapas do desenvolvimento do sistema.
b) Tem como principal objetivo a construção dos casos de uso expandidos e modelo conceitual completo.
c) Deve ser feita de forma relativamente rápida, pois tem como objetivo dar uma visão geral do sistema para o planejamento futuro.
d) Somente pode ser executada após a fase de transição.
e) Não existe tal fase no Processo Unificado.





**2. Assinale abaixo a opção que *não* é uma característica *necessária* de um caso de uso na fase de concepção.**

a) Ser um processo fundamental para a empresa.
b) Ser iniciado e concluído em uma única sessão de uso do sistema.
c) Produzir algum resultado (mudança na informação armazenada).
d) Ter variantes e seqüências alternativas.
e) Ser um processo conduzido por atores, ou seja, entidades externas ao sistema.

**3. Na fase de concepção os casos de uso devem ser ordenados pela sua importância. Qual das opções abaixo é a menos significativa para a definição da importância de um caso de uso?**

a) O quanto o analista pode aprender sobre o sistema com o caso de uso.
b) A complexidade do caso de uso.
c) A ordem temporal de execução dos casos de uso.
d) Quão crítico é o caso de uso para o processo de negócios da empresa.
e) A quantidade de riscos envolvidos no caso de uso.

**4. Considere os requisitos descritos abaixo:**

I. Imprimir relatório de vendas.
II. Registrar pagamento de fatura.
III. Usar banco de dados relacional.
III. Ser compatível com Linux.

a) Apenas I e III são requisitos funcionais.
b) Apenas II e III são requisitos funcionais.
c) Apenas I, II e III são requisitos funcionais.
d) Apenas I e II são requisitos funcionais.
e) Todas as opções acima são incorretas.